EVIM

ENGLISH VALIDATION
&
INTERNSHIP MANAGEMENT

Test Plan

EVIM - English Validation & Internship Management

Riferimento	
Versione	1.1
Data	5/12/2019
Destinatario	Top Management
Presentato da	Edoardo Carpentiero – Attilio Della Greca
Approvato da	Edoardo Carpentiero – Attilio Della Greca

Team Composition

Ruolo	Nome	Posizione	Contatti
Top Manager	Filomena Ferrucci	Rappresentante del cliente	f.ferrucci@unisa.it
Project Manager	Edoardo Carpentiero	Project Manager	e.carpentiero1@studenti.unisa.it
Project Manager	Attilio Della Greca	Project Manager	a.dellagreca5@studenti.unisa.it
Team Member	Simone Auriemma		s.auriemma5@studenti.unisa.it
Team Member	Vincenzo Colacicco		v.colacicco1@studenti.unisa.it
Team Member	Duraccio Michele		m.duraccio3@studenti.unisa.it
Team Member	Giano Antonio		a.giano1@studenti.unisa.it
Team Member	Grieco Simona		s.grieco13@studenti.unisa.it
Team Member	Emilio Schiavo		e.schiavo8@studenti.unisa.it
Team Member	Maria Concetta Schiavone		m.schiavone29@studenti.unisa.it
Team Member	Nicola Sisti		n.sisti1@studenti.unisa.it

Sommario

Revision History

Data	Versione	Descrizione	Autori
25/11/2019	0.1	Impostazione documento	Edoardo Carpentiero Attilio Della Greca
26/11/2019	0.2	Aggiunta paragrafo strumenti testing	Edoardo Carpentiero Attilio Della greca
28/11/2019	1.0	Revisione	Edoardo Carpentiero Attilio Della greca
5/11/2019	1.1	Modifica strategia di testing di integrazione e revisione	Edoardo Carpentiero Attilio Della greca

1. Introduzione

Lo scopo di questo documento è quello di gestire lo sviluppo e le attività di test riguardanti il software English Validation & Internship Management e vengono riportare le strategie di testing adottate, gli strumenti utilizzati e le funzionalità da testare. Il testing è una tecnica di "fault detection" il cui scopo è appunto quello di rilevare errori in maniera pianificata all'interno del codice prodotto. L'obiettivo è, quindi, di evitare che questi errori si presentino in fase di utilizzo del sistema da parte dell'utente finale.

L'attività di testing è una componente fondamentale nello sviluppo di un prodotto software efficiente e funzionale perché permette di rilevare possibili errori all'interno del sistema, che potrebbero verificarsi in seguito alla scrittura del codice sorgente.

In questo documento vengono pianificate e descritte in modo dettagliato le attività di testing che il team si prepara ad effettuare.

Verranno analizzate, in particolar modo, le seguenti attività:

- **Gestione autenticazione;**
- **Gestione tirocinio curriculare;**
- **Gestione proposta tirocinio;**
- **Gestione richiesta tirocinio;**
- **Gestione registro di tirocinio;**
- **Gestione riconoscimento attività lavorativa;**
- **Gestione account.**

1.1 Definizioni, acronimi e abbreviazioni

1.1.1 Definizioni

- **Branch Coverage:** metodo che richiede che tutti i rami del programma o gli stati condizionali vengano testati almeno una volta durante un processo di test;
- **Failure:** mancata prestazione di un servizio atteso;
- Fault: causa di un failure, insieme di informazioni che quando processate generano un fallimento;
- **Model View Control:** modello architetturale comunemente usato per lo sviluppo di interfacce utente che dividono un'applicazione in tre parti interconnesse. Questo viene fatto per separare le rappresentazioni interne di informazioni dai modi in cui le informazioni sono presentate e accettate dall'utente;

1.1.2 Acronimi

- RAD: Abbreviazione utilizzata per indicare il Requirement Analysis Document;
- SDD: Abbreviazione utilizzata per indicare il System Design Document;
- SOW: Abbreviazione utilizzata per indicare lo Statement Of Work;
- EV: English Validation

- IM: Internship Management
- IS: Ingegneria del software

1.2 Riferimenti

- Kathy Schwalbe, "Information Technology Project Management", International Edition 7E, Cengage Learning, 2014;
- Bernd Bruegge, Allen H. Dutoit, "Object-Oriented Software Engineering Using UML,
- Patterns and Java", Third Ed., Pearson, 2010; Sommerville, "Software Engineering", Addison Wesley;
- PMBOK ® Guide and Software Extention to the PMBOK® Guide, Fifth Ed., Project Management Institute, 2013

2. Documenti Correlati

Il presente documento è in stretta relazione con i documenti precedentemente rilasciati, esso è in forte relazione anche con documenti che verranno sviluppati e rilasciati in futuro quindi sarà soggetto a modifiche ed aggiornamenti.
I test case sono basati sulle funzionalità individuate nel documento di raccolta ed analisi dei requisiti.

2.1 Relazione documento RAD

La relazione fra TP e EVIM_RAD_Vers.1.5 riguarda i requisiti funzionali e non funzionali del sistema, in quanto esso contiene la descrizione dettagliata delle funzionalità con scenari, use case, diagrammi e mockup e inoltre, vi è indicata anche la priorità dei requisiti.

2.2 Relazione documento SDD

In EVIM_SDD_Vers.1.1 è presente la architettura del sistema, la struttura dei dati e i servizi dei sottosistemi.

2.3 Relazione documento ODD

In EVIM_ODD_Vers.1.1 sono contenuti i package e le classi del sistema.

2.4 Relazione documento SOW

Nello Statement Of Work uno dei criteri di premialità, è quello di ottenere una branch coverage del 75%, i test case verranno strutturati in modo tale da poter soddisfare tale criterio.

3. Panoramica del sistema

Come definito nel System Design Document la struttura del nostro sistema segue l'architettura MVC (Model View Controller) che espone i servizi utilizzando un approccio REST. La componente fondamentale di questa architettura è il controller. Per ogni sottosistema ci sarà un controller che si occuperà della logica di business della specifica gestione. Nel model verranno mappate le entità persistenti sul DB come oggetti. La view si occuperà di mostrare le interfacce utente. L'approccio REST verrà implementato utilizzando l'oggetto router. Esisterà un router per ogni sottosistema, il quale definirà una specifica URL sfruttando il nome del sottosistema e la firma di un metodo. Presso tale URL verranno esposti i risultati di una interrogazione al DB.

I sottosistemi sono:

- Gestione autenticazione;
- Gestione tirocinio;
- Gestione proposta tirocinio curriculare;
- Gestione richiesta tirocinio curriculare;
- Gestione registro tirocinio;
- Gestione riconoscimento attività lavorativa;
- Gestione account.

4. Possibili rischi

ID	Rischio	Probabilità	Impatto
R16	Pianificazione delle attività di testing non adeguata	BASSA	ALTO
R17	Il team non segue l'approccio definito	BASSA	ALTO
R18	Il team ha delle difficoltà nello specificare i casi di test	MEDIA	ALTO
R19	Il team trova difficoltà nello usare i tool definiti per svolgere l'attività di testing	MEDIA	MEDIO
R20	Le attività di testing proseguono oltre i tempi previsti	MEDIA	MEDIO/ALTO
R21	I casi di test definiti non riescono a portare una branch coverage del 75%	MEDIA	MEDIO

5. Funzionalità da testare e non

Le funzionalità da testare sono tutti i requisiti con priorità media o alta, a differenza dei requisiti con priorità bassa, i quali hanno una rilevanza minore ai fini del test.

- Gestione Autenticazione
  - Login
  - Logout

- Gestione tirocinio curriculare
  - Visualizza elenco tirocinanti
  - Visualizza aziende convenzionate
  - Visualizza elenco docenti disponibili
  - Compila relazione tirocinio
  - Compila questionari valutativi di tirocinio
  - Approvare relazione tirocinio
  - Rifiutare relazione tirocinio

- Gestione proposta tirocinio curriculare
  - Creare proposta di tirocini
  - Visualizzare proposta di tirocinio
  - Modificare proposta

- Gestione richiesta tirocinio curriculare
  - Creare richiesta di tirocinio
  - Visualizzare richiesta di tirocinio
  - Approvare richieste di tirocinio
  - Rifiutare richieste di tirocinio
  - Visualizzare progetto formativo
  - Approvare progetto formativo
  - Rifiutare progetto formativo

- Gestione registro tirocinio
  - Visualizzare registro di tirocinio
  - Compilare registro di tirocinio
  - Approvare attività registro tirocinio
  - Rifiutare attività registro tirocinio
  - Approvare registro di tirocinio

- Gestione riconoscimento attività lavorativa
  - Compilare modulo di riconoscimento attività lavorativa
  - Visualizzare richieste di riconoscimento
  - Visualizzare dettagli richieste di riconoscimento
  - Approvare richiesta di riconoscimento
  - Rifiutare richiesta di riconoscimento

- Gestione account
  - Creare nuovo account

Di seguito sono riportate le funzionalità non candidate al testing perché di priorità bassa:

- Gestione Autenticazione
  - Recupera password
- Gestione tirocinio
  - Annullare svolgimento tirocinio
- Gestione riconoscimento attività lavorativa
  - Modificare modulo di riconoscimento
- Gestione account
  - Modificare dati account

6. Approccio

Sarà praticato uno **unit testing**, ossia si constaterà il corretto funzionamento di tutte le singole unità di codice in determinate condizioni. Sarà quindi testata ogni classe del sistema ed ogni metodo appartenente ad essa.Ogni verifica del funzionamento verrà suddivisa diversi test-case.

Dopo la fase di unit testing seguirà l'**integration testing** con l'obiettivo di collaudare l'interazione tra i vari sottosistemi da cui è composta l'applicazione. Durante questa fase verrà effettuato il **regression testing** per verificare che i nuovi cambiamenti non abbiano introdotto nuovi bug nelle funzionalità di English Validation. Infine, viene effettuato il **system testing** per verificare il corretto funzionamento del sistema completo.

Come strategia utilizzata per effettuare il testing viene utilizzata la tecnica **BlackBox**, che si focalizza sul comportamento Input/Output ignorando la struttura interna della componente.

Per quanto riguarda il testing **WhiteBox** è stato utilizzato il metodo del branch coverage.

Al fine di minimizzare il numero dei test cases, è stato utilizzato il **category partition.**

6.1 Testing di unità

Durante questa fase l'attenzione viene focalizzata sulle singole componenti, in particolare vengono ricercate condizioni di fallimento isolando le componenti tramite utilizzo di Test Stub (implementazione completa o parziale di una componente dalla quale dipende un'altra componente sotto testing) e Test Driver (implementazione completa o parziale di una componente che fa uso di un'altra componente sotto testing). Tali test vengono applicati dopo aver isolato ogni singola funzionalità da testare in modo da verificare una completa validità delle componenti e tener traccia dei possibili errori che vengono riscontrati.
Per tale fase utilizzeremo il tool Mockito per effettuare il testing di unità sulle servlet e JUnit per il testing di unità per le classi.

6.2 Testing di integrazione

Durante questa fase si procede ad inglobare ed integrare tutte le componenti che precedentemente erano state isolate e testate singolarmente, in modo da verificare il corretto funzionamento del sistema anche dal punto di vista globale e non solo dal punto di vista delle singole componenti.

La strategia adottata per il testing di integrazione è quella di tipo "Bottom-up", la quale richiede la costruzione di driver per simulare l'ambiente chiamante e consente di poter iniziare l'attività di testing non appena il primo modulo è stato specificato.

Attraverso tale strategia, vengono testati inizialmente i sottosistemi appartenenti allo strato model dell'architettura software, dopodichè viene effettato il test di integrazione tra lo strato model e controller per poi ultimare con l'integrazione dello strato view.

6.3 Testing di regressione

Man mano che si integrano i vari componenti di IM con il modulo EV, sarà anche necessario rieseguire i precedenti test case di EV. Lo scopo è quello di verificare che le componenti software non influenzate dall'integrazione del nuovo modulo, mantengano lo stesso comportamento anche dopo la modifica.

6.4 Testing di sistema

Prima della messa in esercizio del sistema verrà effettuata una fase di testing di sistema per verificare la corretta adesione dei requisiti individuati con il sistema. Sarà verificata la qualità e l'affidabilità del prodotto software, in modo che possa essere utilizzato dall'utente finale senza margini di errori. Trattandosi di una applicazione web verrà utilizzato il tool Selenium, che si occuperà di simulare l'interazione con il sistema dal punto di vista dell'utente.

7. Criteri Pass/Fail

I criteri determinanti per un **pass** della fase di testing sono l'individuazione di una failure, che si verifica quando l'output osservato sarà diverso dall'oracolo, ossia l'output atteso.

Al verificarsi di una failure si proseguirà con l'analisi dello stesso, in maniera da determinare se questa è legata ad una fault, in tal caso si procederà alla sua correzione. Utilizzando un approccio iterativo, verrà reiterata la fase di test, in modo da avere garanzia di non aver impattato altri componenti del sistema con le ultime modifiche.

La fase di testing avrà esito **fail** se l'output osservato sarà lo stesso dell'oracolo.

8. Criteri di sospensione e ripristino

La fase di testing occupa una parte importante nell'intera attività inerente il progetto, essa è anche molto delicata e può essere causa di imprevisti e slittamenti dei tempi a causa di errori e malfunzionamento del sistema.

8.1 Criteri di sospensione

Il testing sarà **sospeso** quando saranno state testate tutte le classi scelte per ogni possibile input e il risultato sarà quello atteso. Nel momento in cui il testing rivelerà un errore si dovrà passare alla fase di correzione che dovrà essere eseguita da una ripetizione dell'intero processo di testing per rilevare la presenza di eventuali errori introdotti dalle correzioni stesse. Tutto ciò deve essere in accordo con i tempi di sviluppo previsti, tenendo sempre conto dei costi dell'attività di testing.

8.2 Criteri di ripristino

La **ripresa** del testing avviene soltanto quando tutti i problemi relativi alla sospensione dello stesso sono stati risolti. L'attività di testing riprenderà a partire dal test case che ha causato la sospensione.

9. Deliverables

I documenti rilasciati durante e al termine della fase di test sono:

- Test Plan;
- Category Partition;
- Test Case Specification;
- Integration Test Plan
- Unit Test Plan
- Unit Test Report;
- Test Execution Report;
- Test Incident Report;
- Test Summary Report;

10. Strumenti per il testing

Per svolgere le attività di testing è necessario un computer e una copia del DB in locale, in modo tale da non appesantire il DB remoto con dati di prova.
Tool utilizzati per il testing sono:

- **JUnit**: per il testing di unità delle classi
- **Mockito**: per il testing di unità delle servlet
- **Selenium**: per il testing di sistema

11. Necessità di training per il team

Il Team ha effettuato una attività di test durante la parte iniziale del corso di IS quindi conoscono i meccanismi e tool da utilizzare per le varie fasi ti test. Ma essendo comunque la prima volta i PM forniranno al gruppo esempi utili per rendere più chiari e veloci i concetti che non sono chiari.

12. Responsabilità

Ogni team member sarà responsabile del testing della gestione alla quale è stato assegnato. I responsabili generali nonché revisori del Testing saranno **Michele Duraccio** e **Vincenzo Colacicco**.

13. Glossario

- **Developer:** un individuo che costruisce e crea software e applicazioni. Lei o lui scrive,
- esegue il debug ed esegue il codice sorgente di un'applicazione software;
- **Errore**: una misura della differenza stimata tra il valore osservato o calcolato di una quantità e il suo valore reale;
- **Rischio**: risultato potenziale, imprevedibile e incontrollabile di un'azione (o inazione);
- **Tester**: una persona, una macchina o dispositivo utilizzato per verificare se un sistema o componente funziona correttamente;
- **Testing**: processo o metodo per trovare gli errori in un'applicazione o un programma software in modo che l'applicazione funzioni in base ai requisiti dell'utente finale;
- **Tool:** Strumento software utilizzato per ottenere un dato risultato.
- **IS:** Ingegneria del Software
- **IM:**